Life, the Universe, and Everything VDAQ

A Brief Monograph for Experts

or 'What I did on my summer vacation'

by EAH

August 26, 2007 23:33:33 CDT 2007

Disclaimer: The following document is rated 'R' for Reprehensible. It is best viewed by nobody. Due to the dangerous levels of sarcasm and jaded nature of the commentary, the impressionable and/or overly optimistic reader should avert his or her eyes and hum loudly (Disney themes recommended here) if adversely affected or maybe just burn the damn thing outright. The opinions contained within are, of course, those of the author only and do not reflect the views of the University of Chicago, Argonne National Lab, NASA/GSFC, or, well, anybody really.

1 Introduction

In days of yore cartographers marked the bounds of their geographical knowledge with beasts, mermaids, and monsters of the deep[1]. Herein lies my best attempt at summarizing what I know of VDAQ, what I don't know that might be problematic, and beyond that I can only speculate about those regions that may be hiding a beastie. Sarcasm aside, the VDAQ code is a fine and lovely thing and it's been my pleasure to contribute what I can.

The document, or manuscript, if you will, is organized as follows: first, new additions to the code are described in some detail and any known issues, missing features, and the like are listed; second, less significant additions or updates to the code are discussed; third, new scripts are outline; last, known foibles of external systems that have required some action, i.e. software fix, are briefly reviewed. Ahoy, matey! Hic sunt dragones!

2 Significant Additions to the Code

2.1 Configuration Interface to MySQL Database

The New Class on the Block

class VDAQ_VDBInterface

- Member Variables

```
// DB connection member vars
VDBConfig::VDBConfig* fDBConfig;


// FADC/CFD configuration member vars
unsigned int fTelescopeID;
unsigned int fCrateID;
list<VDBFADC::FADCLocation> fFADCInfoList;
```

[1] Here is a lovely dragon from a Japanese map http://www.library.ubc.ca/spcoll/dragon.gif

- Methods

```
// DB connection control methods
void setDBHost(char* hostname) { fDBConfig->setHost(hostname); }
string getDBHost() { return fDBConfig->getHost(); }

// FADC/CFD configuration member functions
void setFADCList(int32_t, int32_t, bitset<MAX_FADC_CNT>, list<uint32_t>);
list<VDBFADC::FADCLocation>  getFADCList(){ return fFADCInfoList; }

// Methods for updating database tables
void updateSlotRelation(int32_t, int32_t, int32_t, int32_t);
void updateCrate();

// Methods for reading configuration information from database
// tables
map<unsigned int,fadc_config_s> loadConfig();
map<unsigned int, unsigned int> loadCFDThresholdsmV(unsigned int);

// Some info output methods
string printCFDConfig(cfd_config_s* cfdConfig);
string printChanConfig(int32_t iChan, chan_config_s* chanConfig);
string printFADCConfig(fadc_config_s* fadcConfig);
```

Comments:

The MySQL code is really straight-forward. Generally, we just call Guy's routines where needed and then place values into our existing configuration structure. The only nice bit for grabbing the CFD thresholds directly is working out the proper query to pull the thresholds

numbers at one go. This takes a little thought about join logic and looks pretty nasty written out, but can be easily tested using the VDBtools or even better by just mucking with the mysql command line a bit. Almost any mysql query you'll ever want or something very similar to it exists in the VDBTool code at this point. The only thing obnoxious about this stuff is setting up an interface for the query while allowing all those troublesome user options.

Writes to the current configuration settings are slow because they are handled internally line-by-line because of the way error checking and timestamping is done in VDBlib. What I mean is that a straight insert of 500 rows of data is pretty fast, but updating and adding 500 lines one-by-one is, of course, slower. I expect there are clever ways around this; however, it never seemed worth the time since updating the hardware settings isn't really something the observers need to have happen on a dime. I'd prefer they think before they set.

Gotchas Of Note:

- **The use of VDB requires environment variables**

  There are elegant ways to handle environment variables when running remote jobs, and then there is the way that I did it, i.e. brute force. They are currently set explicitly in the runner script. My preferred solution of setting the VDB configuration internally to the software fails on the crates, and I didn't spend the time to fully debug the details. It is possible this scheme could still work by handling the config setting in a more static sense, but it is also equally possible that there is something inherent in the way VDBlib constructs the connection that is screwing this up. Anyway, went beyond my expertise and interest - too bad.

- **Auto-update of crate ID map disabled**

  We used to dynamically update the crate maps in the database on start-up, which was handy, but no more! Due to the flaky FADC board ID problems, this has been rather crudely disabled. The code is still in place, but you really get into trouble fast if you

let VDAQ load wrong ID maps.

- **Ongoing VDAQ uglification program**

  I inherited the core code for pulling down the configs from Guy and didn't rewrite it initially. For reasons of maintenance, I did later make some attemt to clean this up and standardize it to the rest of the VDAQ code, but I only got it partly tamed.

- **VDB has been lizified**

  Ahm...I will be committing that extra method that only exists in my version of VDBlib to the CVS repository ASAP, in fact, probably even before I send you this – only breaks the VDBtools if someone actually updates VDBlib on dacq anyway.

Of note — Not!! but interesting all the same

The name of this class is a bit of a stretch; this is not a true C++ interface class because it contains data members. In a related aside, I'm not actually *that* geeky. The only reason I became aware of this was because I extended the VME functionality[2] for the TRICE code by using an interface class to register VME modules requiring special handling on exit with the VME interface. This extension is useless for Veritas, but it was a kind of neat way to make the shutdown flexible for various VME modules.

Also, you may note there are two constructors for the class, and they are not compatible. Only one applies to VDAQ FADC class members. This makes sense when you realize it's quite handy to develop and test most of the DB crap independently of vme libraries and the rest of the VDAQ code.

[2]P.S. This is irrelevant for vdaq, but TrICE exposed a couple minor bugs in VME interrupt handling. I doubt I got around to putting my fixes in vdaq since this struck me as code you would revamp if it got used much. There is a bug in the VME interrupt handler that allows interrupts to arrive before the count has updated. My solution was to bump up the count prematurely and bump it down if vme_interrupt_attach failed. Also the killInterruptHandler method should decrement the handle count in this scheme.

2.2 Bypass (fast-like-tha-Freeway) Threshold Setting

This is a little known but critical and yet easy kludge with only one minor gotcha. I know that you know this stuff, but for the sake of clarity, I'm gonna set it all out. Setting the thresholds is dog, and by that I mean old-yeller houn' dog, slow. Here is why:

How to set CFD thresholds (most robust)

1. User selects mV threshold
2. Grab list of CFD IDs from the FADC Location and FADC CFD Relation tables
3. Fill CFD Settings for each CFD ID from respective Settings tables
4. Assign mV setting for each CFD
5. Convert physical to dc settings for each CFD Setting [3]
6. Write timestamped dc settings to CFD Threshold Settings table (most time is spent on the loading step)
7. Send command to VDAQ to read config settings
8. VDAQ reads and applies configurations
9. VDAQ returns Success/Fail to eventbuilder (no fail states defined yet...)

There are a couple steps there that we really do not need given CFD coefficients don't change much and for the special case of bias curves.

How to set CFD Thresholds for Bias Curves

1. User selects mV threshold
2. mV Threshold passed to VDAQ via the eventbuilder

[3] This step is skipped in the current vdbtools, and the lookup table is queried instead.

3. VDAQ grabs dc thresholds for all CFDs in crate from lookup table

4. VDAQ loads CFD thresholds

5. VDAQ returns Success/Fail to eventbuilder (no fail states defined yet...)

IMPORTANT NOTE!!! The DB CFD Settings are not updated when this happens. Users are not given an option to set thresholds this way (unless I was ignored.) Of course an expert can shortcut via the eventbuilder... ;-) VAC must end all bias runs by telling VDAQ to load configurations from the database to ensure everything matches up.

Code Changes:

- SCI commands defined for passing a mV threshold request and returning status.
- New routines in various FADC classes to handle retrieving and setting the CFD thresholds for crates and boards via the DB.
- The physical units option no longer makes sense with the DB configs and can actually cause some nasty problems. I've disabled it and left some warnings, but you might consider removing it.

Of Note

It is, of course, easy enough to adapt the current routines to set values across the array. Sorry I didn't get around to this before taking off, but I do have ulterior motives. If the VDB tools retain flaws, ideally in the interface, then some useful person will take it up to transfer the code into VAC as was intended. Actually, a great second-wave project for VAC+Guy is speeding-up the VDB methods a notch and parallelizing this sort of access, assuming the DB can handle it sensibly - it should or definitely could.

Gotcha: Keeping the lookup table up-to-date

This isn't strictly a VDAQ issue, but you should be aware of it. The lookup thresholds are obtained from a precalculated table in the database generated by me. If the coefficients for

a CFD change or new CFDs are introduced into the system, the lookup table will need to be recalculated for those IDs. This shouldn't come up much, and I have already generated the lookup table for known CFD IDs.[4] In a perfect world, the routines that handle loading CFD coefficients or assigning CFD IDs to boards would check this. I have a routine that verifies existing settings and calculates the conversions. This can be run periodically without ill effect to be sure that things line up. SJF's diagnostics catch CFDs that aren't being programmed properly so we can't get into too much trouble.

3 Insignificant Additions to the Code: Known Kludges

- **Flaky Magic Serial Signal Register Reads**

  We read the event mask registers something like 5 times in a row to make sure we get it – i.e. this is really just a dumb delay. I don't even know if this is still a problem. I haven't gotten around to looking at it again since bringing up the skeleton system.

- **FADC boards go mental, forget ID**

  What is up with this? I don't know what is causing it, but I added some fairly obnoxious warnings in the DBInterface class to catch board ID screw-ups and to alert the observers.

- **Bad event masks/noise triggering**

  Once again, we clean up the messes of others. When the DATs screw up the event mask or cause noise triggering, event IDs look wacky and trigger IDs are nonsensical. The code checks for these and complains loudly.

[4] I cover thresholds from 20 mV to 300 mV in 1 mV steps, and VAC knows to respect these bounds in the bias curves, but leave it to some moron...

4 OMG Scripts!!!

I think I've mangled every script we have in some fashion. A couple of them are new to the array and some are pretty recent. Let's go reverse chronologically on the last touch by me. All my scripts are available at www.hep.anl.gov/ehays/vdaq-scripts.

update-vdaq-all	Apr. 2007	dacq-t1 ∼/liz/vdaq-scripts
Purpose: run svn update on all 4 dacq machines and report anything egregious. If update is successful, builds VDAQ. Records output locally for easy perusal.		
stopVDAQ.pl	Jan. 2007	arrayctl /usr/local/veritas/bin
Purpose: run vdaq-end-night on specified telescopes from arrayctl Known Issues: Observers are not fans of the reporting and neither am I.		
SetVDAQDisk.pl	Dec. 2006	VME crate disks /13
Purpose: Alters all necessary info to identify a newly cloned or existing disk with a given telescope and crate. Does not handle the mysql libs since that is dacq dependent and is more of an update than an install function.		
vdaq-end-night	Nov. 2006	dacq-tX ∼/run-scripts/
Purpose: Halts crates, syncs logs to dacq, and powers down crates. A possible flaw is that this script and sub-scripts have telescopes hardcoded. Granted a one-line perl command will update them to a new telescope. Sub-scripts are fairly obvious, Stop_VDAQ.pl, Copy_Logs.pl, and Halt_Crates.pl.		

5 Unresolved Bits and Bots

5.1 State of the Repository

We are using the T2 branch as our deployment version everywhere. The last time I was in AZ[5] I synced up the code amongst all the telescopes and resolved any significant differences.

[5] Gawd! has it been that long...getting telescope withdrawal shakes just thinking about it...

I put together a simple bit of script www.hep.anl.gov/ehays/vdaq-scripts/update_vdaq_all to do array-wide updates. It logs info from all scopes to the directory it runs in. (hm. timestamping these would have been handy; I overwrite previous info. oh well.) and parses out any reported conflicts. I believe there have been only two software changes since then.

- T3:: main aux code:: fix that doesn't seem to fix much relating to the GPS clock.
- T1:: main :: disabled VME reset via SCI, but I recall undoing this later?

Unfortunately for you and me both, I am not much of a code czar and am in remiss for not maintaining things a bit more tidily. Tags of major revisions would have been kind of nice in hindsight.

5.2 VDB/MySQL library versions on dacq and crates

Although all the telescopes are 'the same', they are not exactly the same as you're well aware. The most obnoxious difference relating to my stuff is that each RH install on the dacq machine ends up with a different version of the mysql client libs. The libs on the crates should match those on dacq and can be copied over. If VDAQ is hanging on the initial configuration download, this might be the reason. Another problem encountered on install is that db.vts should be in the hosts file. Jeremy tends to wipe out the hosts file and replace with the dns, but I like to keep IPs relevant to VDAQ hardcoded since they are very limited in number and don't tend to change. Also, the dns machine in the past was sometimes flaky and the last thing I want to debug for the observers is some *terrible* and *catastrophic* problem in vdaq that turns out to be some flaky dns machine. I had the vague goal that vdaq should be as autonomous a system as possible and should still be runnable in the absence of the network/VAC/dacq/db/etc. for purposes of basic debugging and whatever the future may hold.

My attitude toward the VDB lib was to never update it. Nothing related to FADC/CFD routines has changed in years now.

5.3 VDBTools are not included in VDAQ

I have kept these independently of VDAQ for obvious reasons and consider this a feature. They fall outside the bounds of the subproject, and I figure all the better to keep them as separated and outlawlike as possible to encourage later absorption into a more fitting home. All documentation is in the code or the wiki – will update this a bit in the next week to prep for observing.

5.4 FADC-related 'obnoxieties'

The FADC boards are all the same revision. Except when they are not.

- All the early board IDs have been retrofitted to the latest revision. Previous adjustments for FPGA revisions in our software, used on Board IDs $< 6/< 28$, have been disabled. I believe that I propagated this change to all telescopes, but it is something to consider if a newly swapped low number board looks wacky. (Newly redone boards only appear on T3/T4 at the moment, and so I'll only swear to those.)

- The ring buffer in the latest revision boards (new for T4 and redone prototype) is 16 kB and not 32 kB.

- Jim will swear up and down that the high multiplicity trigger works and is just sitting there being ignored by evil old us, but actually Paul disabled this functionality (perhaps to fix CBLT/ZS timing probs?). Jim and Amanda looked at this circa. Feb. 2006 and found there was no high multiplicity output being generated by the C/T boards.

5.5 A many-windowed VDAQ

I know there are occasional grumblings about all the VDAQ windows. I almost swutched this over in Feb. 2006. At the point where I considered going headless and would have had time to tweak it, I had to back off due to too many dumb problems with DATs/FADCs/trigger/crate

drops/you name it. If we didn't end up debugging every dang thing both up and downstream of us, this might be more feasible. The observers tend to ignore things even more than normal without the visual feedback and are more likely to jump to the wrong conclusions when things go 'pear-shaped'. Clearly we're not going to stream our lovely error messages to VAC. Granted, with some devoted VAC work, we could set up some sensible error codes in a fairly rational way. It wouldn't be too bad on certain error conditions to send an SCI ping to dacq. However, it doesn't really make sense to try to define standard error codes until the DAT situation is resolved or it is clear that there won't be any reoslution anytime soon. The real problem is that this kind of update requires a VDAQ savvy person hanging around for a bit and getting the bugs worked out. Unfortunately, we didn't hit semi-stable operation before I got busy. Even the problem of of wacky fail states could be countered with some creative error checking and a centralized logging interface – oh well.

6 Conclusion

So that's all she wrote. I've highlighted the bits of code I've most impacted and laid out some of the remaining issues as I see them. I have a couple outstanding issues to fix in the VDBtools – I owe Michelle Hui a tool for loading the laser lookbacks – but that's not your problem. As I said, I'll give a run through the documentation on the Wiki relating to my bits this week and get it up to spec. Have fun.

Vivat Veritas Hephaestus gratia.